



Reinforcement-learning generation of four-qubit entangled states

Sara Giordano ¹ and Miguel A. Martin-Delgado ^{1,2}

¹*Departamento de Física Teórica, Universidad Complutense, 28040 Madrid, Spain*

²*CCS-Center for Computational Simulation, Campus de Montegancedo UPM, 28660 Boadilla del Monte, Madrid, Spain*



(Received 26 May 2022; revised 3 October 2022; accepted 6 October 2022; published 25 October 2022)

We have devised an artificial intelligence algorithm with machine reinforcement learning (Q-learning) to construct remarkable entangled states with four qubits. This way, the algorithm is able to generate representative states for some of the 49 true SLOCC classes of the four-qubit entanglement states. In particular, it is possible to reach at least one true SLOCC class for each of the nine entanglement families. The quantum circuits synthesized by the algorithm may be useful for the experimental realization of these important classes of entangled states and to draw conclusions about the intrinsic properties of our universe. We introduce a graphical tool called the state-link graph (SLG) to represent the construction of the quality matrix (Q-matrix) used by the algorithm to build a given objective state belonging to the corresponding entanglement class. This allows us to discover the necessary connections between specific entanglement features and the role of certain quantum gates, which the algorithm needs to include in the quantum gate set of actions. The quantum circuits found are optimal by construction with respect to the quantum gate-set chosen. These SLGs make the algorithm simple, intuitive, and a useful resource for the automated construction of entangled states with a low number of qubits.

DOI: [10.1103/PhysRevResearch.4.043056](https://doi.org/10.1103/PhysRevResearch.4.043056)

I. INTRODUCTION

The term machine learning (ML) was coined in 1959 by Arthur Samuel, an American IBMer and pioneer in the field of computer gaming and artificial intelligence [1]. ML consists essentially in the exploitation of particular algorithms of self-learning to get information from data in order to make predictions. Because of this, machine learning facilitates computers in building models from sample data in order to build decision-making processes based on data inputs [2,3].

The applications of machine learning techniques to physics experienced a huge development in the last decades, with approaches based on topological optimization, evolutionary strategies, deep learning and reinforcement learning [4–6]. In particular, the application of ML to “creative tasks,” such as designing new quantum experiments, is not completely explored yet. There are a few aspects of quantum mechanics which lead researchers to think that ML and computer aided techniques offer interesting and promising perspectives in this regard [5]. For example, as we start to deal with many entangled particles, the Hilbert space dimension becomes so large that the problem is no more treatable without computer aid. Moreover, to build new experiments, physicists usually have to handle a huge number of variables, and ML techniques are yet in use to handle this kind of complexity [4]. Moreover, it is worth asking whether ML and artificial intelligence (AI) can boost human intuition in dealing with the intrinsic counter-intuitive nature of quantum mechanics, and if ML can also

help to handle multidimensional and multipartite entangled systems. An emblematic example is the Melvin algorithm developed by Krenn *et al.* in 2016 [6]. Indeed, this algorithm has uncovered solutions to previously unsolved questions and has also inspired the discovery of new scientific insights [7].

In this work, we will present an application of a ML technique to quantum entanglement, a fundamental resource for quantum computation. In particular, we will employ ML to design new quantum experiments aimed at engineering quantum mechanical states with desired entanglement properties.

II. BACKGROUND

Machine learning algorithms can be grouped into three basic types: Supervised learning, unsupervised learning and reinforcement learning (RL). Many ML reinforcement methods, such as the projective simulation (PS) algorithms [8], have been extended to the quantum domain [9] obtaining a quantum advantage for the first time over their classical counterparts. Whereas a lot of activity is being generated for quantum reinforcement learning [10,11], the present works remains classical with respect to the algorithm employed with the goal being the generation of quantum states. In particular, we will employ an RL algorithm called Q-learning. This latter is one of the best known RL algorithms [3,12], and, although the name may suggest otherwise, the Q does not stand for anything quantum-related, it is in fact a historical notation based on the name of the cost function to be optimized during the algorithm.

A. Q-learning algorithm working principles

Q-learning, like all the RL procedures, involves an agent (the algorithm itself), an environment and a set of actions by which the agent interacts with it (see Fig. 1). In the

Published by the American Physical Society under the terms of the [Creative Commons Attribution 4.0 International license](https://creativecommons.org/licenses/by/4.0/). Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

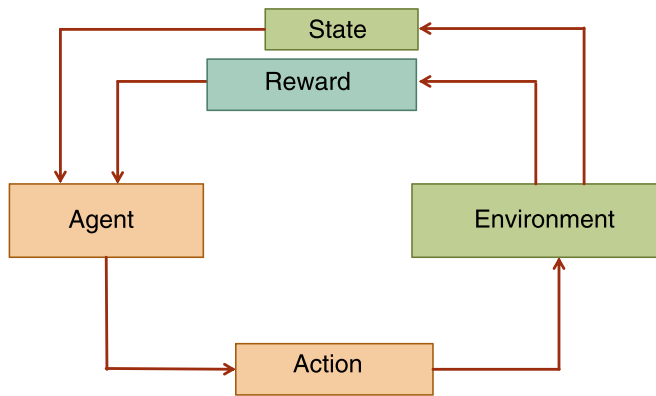


FIG. 1. Flowchart of reinforcement learning working principle. It consists of an agent, an environment and a set of actions by which the agent interacts with the environment and is rewarded.

environment are placed some *objectives* which the agent must reach [13]. It is a policy-free RL algorithm, i.e., there is no previous knowledge of a conduct of behavior [12]. It can explore the environment by performing stochastic actions and analyzing the feedback it receives. By interacting with the environment, the agent changes its state and eventually earns some rewards, when finding its objectives. By keeping track of this feedback, it learns how to interact with the environment to reach the objectives while maximizing the rewards. Hence, the agent will be able to select its actions with an optimal policy [12], in the sense of the reward maximization.

A simple illustrative description of the Q-learning algorithm is shown in Fig. 2 for the mouse-labyrinth example. We can identify the following elements: (1) Ensemble of actions: all the possible movements that the mouse can perform, (2) states of the agent: Its positions in the labyrinth, and (3) rewards: The pieces of cheese which he can reach while walking in the labyrinth.

The actions of the mouse (agent), allows it to explore the labyrinth (environment). The states in which the mouse

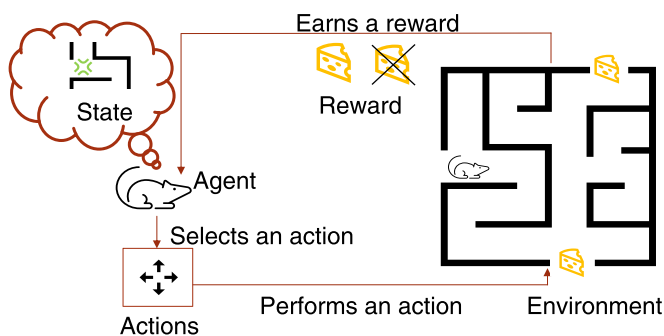


FIG. 2. Q-learning scheme through the mouse-labyrinth example. In this image, we represent the agent as a mouse that can move around a labyrinth, with the aim of finding the exits, where some cheese-rewards are placed. In the learning process, it selects and performs random movements starting from random positions in the labyrinth. It eventually earns a reward when it approaches the escapes and records this event. By performing random movements and by keeping track of the rewards earned, it can trace the optimal path to escape from the labyrinth, which maximizes the rewards earned.

could be corresponds to all the positions in the labyrinth. The rewards in the environment are encoded by a user defined function, which establishes the “prizes” that the agent will gain during its exploration. Usually this means that, if we want the agent to learn how to end up in specific states in the environment, we have to provide rewards every time it performs actions which lead it in those states. We will call these states the “objective states.” The reward function is usually implemented as a matrix called Reward matrix (R-matrix). It has nonzero elements for *state-action* pairs that lead directly to the objective states.

While the agent explores the environment, it needs to record the rewards it earns and to recall them during the exploration. In particular it needs an instrument to keep track of the pairs *state-action* which bring to rewards. In Q-learning this tool is the Quality matrix, or Q-matrix. The Q-learning algorithm involves two phases: Training and testing [12]. The training part consists in the exploration of the environment and proceeds with single *episodes*. At each time t an episode takes place, the agent is placed in a random state s_t in the environment, from which it performs a random action a_t . It records the reward which it eventually obtains from the R-matrix, R_t . The numerical value inserted in the Q-matrix is calculated with a Bellmann equation [14], or Q-learning formula, which allows to update the old Q-matrix value $Q(s_t, a_t)$ to the new one $Q^{new}(s_t, a_t)$. The agent assigns to each pair *state-action* a quality value (Q-value), which depends not only on the reward earned in the current episode (R_t), but also on the rewards received in the past ones. Further details about the calculation of this Q-value will be given in the following section. Below are shown the steps consisting in a single episode:

```

    =====
    Q-learning - episode of the training part
    =====
    Initialize: Generate a random state  $s_t$ 
    Action : Select a random action  $a_t$ 
                Perform the action  $a_t$  from the state  $s_t$ 
    Observe: Check the resulting state  $s_{t+1}$ 
    if  $R_t \neq 0$  then
        Earn a reward
    end
    Update  $Q(s_t, a_t) \rightarrow Q^{new}(s_t, a_t)$  with Bellman eq.
    Initialize: Generate a new random state
    =====
    
```

At the end of the training part, the agent has updated its Q-matrix with weighted rewards associated with the pairs *state-action*. Each value in the Q-matrix quantifies how much is *good* to take a specific action in a certain state, in order to reach the objective with an optimal path.

If the agent went through a sufficiently high number of episodes, the values of its Q-matrix no longer change significantly. The criteria for this convergence must be established depending on the problem treated, but in general it is theoretically proven [12,15] that the Q-learning algorithm converges to an optimal policy. In the testing part, by using the Q-matrix values, the agent is able to reach the *a priori* established objectives by following the best rewarded pairs *state-action*, starting from a chosen initial state. The agent selects from the Q-matrix the most rewarded action associated with its current state. It performs the action and changes its state, repeating

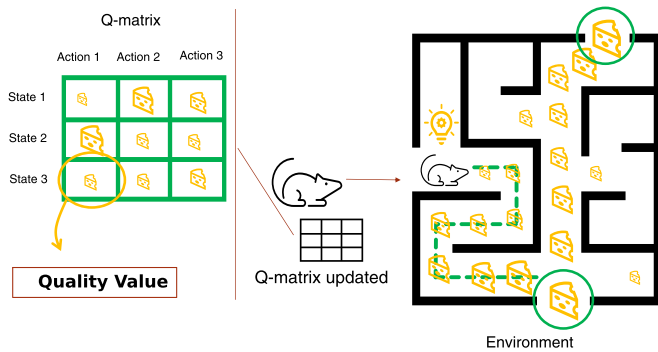


FIG. 3. At the end of the training part, the mouse has updated its Q-matrix with weighted rewards associated with the pairs *state-action*. These values quantify how much is *good* to take a specific action in a certain state, in order to reach the objective. In the grid showed in this picture, the size of the cheese values indicates the *goodness* of an action, given the state in which it is performed. Once an initial position is established, the mouse follows his grid of *state-action* pairs to select his next movement, choosing the most rewarded ones. This will carry it to the exit of the labyrinth.

this procedure until it reaches the final objective. In this way, it can find the optimal path towards its goals, maximizing the earned rewards (see Fig. 3).

We choose this algorithm because, even if the problem of designing quantum circuits is a deterministic problem, we need to define a code of conduct that establish which action to choose in each state, or that establish the parameters that we should evaluate in order to select an action (i.e. we are missing a policy or policy evaluation). Because of that, we chose to avoid value iteration procedures [16], or other dynamic programming methods. Moreover, we did not introduce a distance concept in our environment, and thus we are not able to assign values to the states based on their distance from the objective state.

B. Q-learning cost function

As we mentioned, the Q-matrix is derived from the R-matrix, and it is updated at each episode of the training process using the Q-learning formula:

$$Q^{\text{new}}(s_t, a_t) \leftarrow Q(s_t, a_t)(1 - \alpha) + \alpha(R_t + \gamma \max_a Q(s_{t+1}, a)). \quad (1)$$

This expression is a Bellman equation [14], where $Q^{\text{new}}(s_t, a_t)$ expresses the quality value of a *state-action* pair. In this element, the variable t represent a discrete timing of the episodes. We can compute the value $Q^{\text{new}}(s_t, a_t)$ using the prior value of the Q-matrix for that *state-action* pair, $Q(s_t, a_t)$, and adding to it the reward earned in the current episode, R_t . We add to this latter an evaluation of the maximum reward for the possible future actions, $\max_a Q(s_{t+1}, a)$, in fact, the a index runs through all the actions that can be performed from the resulting state s_{t+1} . If in the past episodes the agent gained some rewards it will take into account these past rewards as a positive contribution to the $Q^{\text{new}}(s_t, a_t)$ value, thanks to the quantity $\max_a Q(s_{t+1}, a)$. This means that, in order to assign a quality value to the current *state-action* pair, the agent is looking at the values of the possible next steps. Indeed, with

this procedure, the agent learns to follow the best rewarded path. These past rewards are scaled by a *discount factor* γ .

(1) γ : Is a real number between zero and one, $0 < \gamma < 1$, and sets how much the reward of future actions influences the new value $Q^{\text{new}}(s_t, a_t)$.

If γ is chosen close to 0, it will prevent the algorithm to see the future rewards by making it “myopic” [12], considering only current rewards. If it is close to 1, the past rewards will be taken much more into account than the new ones coming from the R-matrix. All these terms are scaled again by the so called *learning rate* α .

α : It is a real number between 0 and 1, like γ , it is set to establish the learning speed of the algorithm.

If it is closer to 1 it allows the algorithm to learn quickly, as the values of the Q-matrix will be updated with a high weight, otherwise, the learning part of the algorithm is slower but is capable to “remember” better the rewards taken in the past. In other words, α defines how much we override the old values of the Q-matrix with the new ones. Hence, the setting of α requires to choose between two opposite strategies. The first, with α close to 1, consists in a faster exploration, which easily forgets the past rewards, in favor of the new ones. The second, with α close to 0, implies a slower exploration in which, however, the algorithm can remember each past reward, but it will take a longer time to explore the environment. After setting those parameters to suitable values, the algorithm will update the Q-matrix during the training part.

The Q-values are proven to converge to those of a $Q^*(s, a)$ function, which represents the optimal policy of the algorithm, i.e., the set of quality values that make the agent able to reach the objective with an optimal path [12]. However, the training part should include ideally an infinite number of trials in order to convergence to the optimal policy [12,15], in practice, there is a residual convergence, which is negligible. We will consider the training completed when the values of the Q-matrix remain stable under a certain threshold.

C. Entanglement basics

Entanglement is one of the most important and interesting features of quantum mechanics, it has a key role in the fields of quantum communication, quantum cryptography and entangled states are a fundamental ingredient to build quantum algorithms in quantum computation [17,18]. For pure quantum states we can say that if we have two or more systems entangled the state of each system cannot be described independently from the others. In other words, for a system with Hilbert space $H = H_A \otimes H_B$, it holds

$$|\psi\rangle \neq |\psi\rangle_A \otimes |\psi\rangle_B, \quad (2)$$

with $|\psi\rangle$ being an entangled state of the whole system in H , $|\psi\rangle$ cannot be factorized into the states of the subsystems A and B . We focus our discussion on two-level quantum systems, i.e., the qubits (such as linearly polarized photons or electrons spin), which generic state reads

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad (3)$$

with α and β complex coefficients. We can deal at the same time with more than one qubit, and thus we can have multi-qubit entanglement.

The study of entanglement between qubits is of crucial importance for quantum computing and of course for the realization of modern prototypes of quantum computers [17,18]. Many studies and applications have been made to obtain desired quantum states, with particular entanglement properties, and the entanglement complexity poses major obstacles to this research field [5]. For this reason, many recent works employ ML techniques in order to help scientists in difficult computational tasks, but also to explore entanglement features, which are far from human intuition [4,6]. With such background, this work is devoted to explore the design of quantum circuits that can reproduce entangled states of qubits, with the aid of a Q-learning algorithm. In particular, we will focus our attention on entangled states of four qubits, based on the classification made previously in the literature [19–23].

D. SLOCC classification

The complexity of quantum entangled states requires a clear picture that allows to classify them. In order to categorize different types of entanglement, we can divide the Hilbert space of a multipartite system into equivalence classes, using an operational definition of equivalence. Following the scheme in [24,25] we can use the Local Unitary (LU) equivalence, with LU being deterministic and reversible operations. If it is possible to transform a state into another via LU operations, then the two states are LU-equivalent [25]. Two LU-equivalent states have the same physical properties, in particular, the same entanglement ones. However, the LU operations do not include all the possible operations that preserve the entanglement and can be performed experimentally. In particular, they do not allow to perform joint operations on spatially separated particles. In order to classify properly the entangled states, we need to include in the conversion operations the support of classical communication [26]. This leads to the paradigm of Local Operations assisted with Classical Communication (LOCC): quantum states are transformed by performing Local Operations (LOs) on the subsystems and allowing the transmission of classical communication between the spatially separated parties [25,26].

For the case of pure states, however, it has been shown [27,28] that two states are LOCC-equivalent iff they are LU-equivalent. This means that the classes defined by LOCC are the same as those defined by LU operations. It has been demonstrated that two pure bipartite states are LOCC-equivalent iff they have the same Schmidt coefficients [19,29,30]:

$$|\psi\rangle \stackrel{LU}{\leftrightarrow} |\phi\rangle \Leftrightarrow |\psi\rangle \stackrel{LOCC}{\leftrightarrow} |\phi\rangle \Leftrightarrow \alpha_i = \alpha'_i, \quad \forall i, \quad (4)$$

with $|\psi\rangle \rightarrow \{\alpha_i\}$, $|\phi\rangle \rightarrow \{\alpha'_i\}$ being their Schmidt decompositions. Hence, through these LOCC we can transform one state into another deterministically (with probability of success equal to 1), i.e., the two states belong to the same LOCC entanglement class [31], and they have the same entanglement properties. Otherwise they belong to different entanglement classes and thus have different entanglement properties. We notice that this criterion is interesting in quantum information theory because all the parties involved can use these LOCC equivalent states for exactly the same tasks [31].

This Schmidt criterion of classification becomes impractical when dealing with multipartite Hilbert spaces, in fact, the

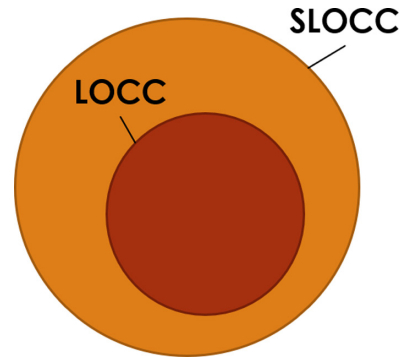


FIG. 4. LOCC classification as a refinement of SLOCC classification. States which are LOCC equivalent result also SLOCC equivalent [25].

Schmidt decomposition is only possible for the bipartitions of a system [17]. The most promising classification for multipartite Hilbert spaces states is the one based on the equivalence under Stochastic Local Operations and Classical Communication (SLOCC). This latter is identical to LOCC equivalence except that the interconversion of two states does not need to be deterministic, the success probability of a conversion only needs to be nonzero [17]. Thus LOCC-equivalence implies SLOCC equivalence, and therefore the partition of the Hilbert space into LOCC equivalence classes is a refinement of the partition into SLOCC classes, Fig. 4. The SLOCC classification is carried out mostly by means of invariants and semi-invariants [23,25]. We remind that SLOCC operations cannot increase, on average, the amount of entanglement [17,25]. In particular, it is not possible to generate entangled states from separable states by SLOCC, even probabilistically [17,25].

With two qubits, there exist only two SLOCC equivalence classes, the class of separable states, and the class of entangled states. Any pure entangled state of two qubits can be converted via SLOCC into any other pure entangled state with nonzero probability. For three qubits, there exist six SLOCC classes [31], namely, the separable class, the three bipartite entangled classes AB-C, AC-B, BC-A, the class with W-type entanglement and the class with GHZ-type entanglement [31,32].

For four qubits, the number of SLOCC classes seemed to become infinite [31]. This assumption was then disproved in [25] by D. Li *et al.*, in particular they specify that it cannot be asserted that there exist infinite SLOCC classes for four qubits. One of the reasons lies in the fact that SLOCC classes of n qubits depends on at least $2 \times 2^n - 2 - 8n$ real parameters [21], this lower bound allows for a finite number of SLOCC classes for $n = 4$ [25] in the SLOCC representation made by F. Verstraete *et al.* Meanwhile, various attempts have been made to find physically meaningful classification schemes for the four-qubit case. The technique that we will take into account is the one used in Refs. [20,21], developed also in Ref. [23] and then further analyzed in Ref. [22]. F. Verstraete *et al.* [21] introduced the concept of Entangled Families (EF), and nine different EFs were found, while in Ref. [22] 49 different SLOCC classes are identified within these EFs. This classification of four qubits states will be the basis for the application of our quantum circuit design algorithm.

E. SLOCC families for four-qubit entangled states

As we explained in the previous section, Verstraete *et al.* categorize the four-qubit entangled states into nine entanglement families basing the categorization on SLOCC

classification [21]. They prove that each SLOCC equivalence class belongs to exactly one EF [21]. In Fig. 5, a graphical representation of this relationship is shown. Here we report all the nine EFs for four qubits, we adhere to the terminology used in Ref. [21]:

$$\begin{aligned}
 G_{abcd} &= \frac{d+a}{2}(|0000\rangle + |1111\rangle) + \frac{a-d}{2}(|0011\rangle + |1100\rangle) + \frac{c-b}{2}(|0110\rangle + |1001\rangle) + \frac{b+c}{2}(|0101\rangle + |1010\rangle), \\
 L_{abc_2} &= \frac{a+b}{2}(|0000\rangle + |1111\rangle) + \frac{a-b}{2}(|0011\rangle + |1100\rangle) + c(|0101\rangle + |1010\rangle) + |0110\rangle, \\
 L_{a_2b_2} &= a(|0000\rangle + |1111\rangle) + b(|0101\rangle + |1010\rangle) + |0110\rangle + |0011\rangle, \\
 L_{ab_3} &= a(|0000\rangle + |1111\rangle) + \frac{a+b}{2}(|0101\rangle + |1010\rangle) + \frac{a-b}{2}(|0110\rangle + |1001\rangle) + \frac{i}{\sqrt{2}}(|0001\rangle \\
 &\quad + |0010\rangle + |0111\rangle + |1011\rangle), \\
 L_{a_4} &= a(|0000\rangle + |0101\rangle + |1010\rangle + |1111\rangle) + (i|0001\rangle + |0110\rangle + i|1011\rangle), \\
 L_{a_20_{3\oplus\bar{1}}} &= a(|0000\rangle + |1111\rangle) + (|0011\rangle + |0101\rangle + |0110\rangle), \\
 L_{0_{5\oplus\bar{3}}} &= |0000\rangle + |0101\rangle + |1000\rangle + |1110\rangle, \\
 L_{0_{7\oplus\bar{1}}} &= |0000\rangle + |1011\rangle + |1101\rangle + |1110\rangle, \\
 L_{0_{3\oplus\bar{1}}0_{3\oplus\bar{1}}} &= |0000\rangle + |0111\rangle,
 \end{aligned}
 \tag{5}$$

where $a, b, c,$ and d are four complex parameters. The SLOCC classes which can be identified in these nine families depend on constraints applied to those four complex parameters [22]. The work by D. Li *et al.* [22] distinguishes at least 49 true SLOCC entanglement classes among all the nine families. For example, for family G_{abcd} they identify 13 different true SLOCC classes; for family L_{abc_2} , 19 true SLOCC classes and so on. They give the complete SLOCC classifications for families $L_{a_4}, L_{a_20_{3\oplus\bar{1}}}, L_{0_{5\oplus\bar{3}}}, L_{0_{7\oplus\bar{1}}}$, and $L_{0_{3\oplus\bar{1}}0_{3\oplus\bar{1}}}$, but they do not grant that the other classes have been completely explored. In Table I, we summarize the 49 true SLOCC classes, specifying the conditions on the coefficients a, b, c and d . Here we adhere to the terminology used in Ref. [22].

Notice that, for $L_{a_20_{3\oplus\bar{1}}}, L_{0_{5\oplus\bar{3}}}$, and $L_{0_{7\oplus\bar{1}}}$ in Eq. (5), there is a 1 : 1 correspondence with SLOCC classes, while $L_{0_{3\oplus\bar{1}}0_{3\oplus\bar{1}}}$ does not contain four qubit entanglement, in fact it is a product state of the one-qubit state $|0\rangle$ and the three-qubit GHZ state. It is clear that, with the increasing number of qubits, the number of different SLOCC classes increase dramatically, i.e., the complexity of the entanglement grows exponentially with the number of qubits involved. To overcome this complexity and approach entangled states from a computational point of view, we decide to apply the RL algorithm described in Secs. II A and II B, called Q-learning [12], in order to synthesize quantum circuits which can produce four-qubit entangled states belonging to the above showed EFs. We will illustrate in detail in the next sections how Q-learning is exploited for our purpose.

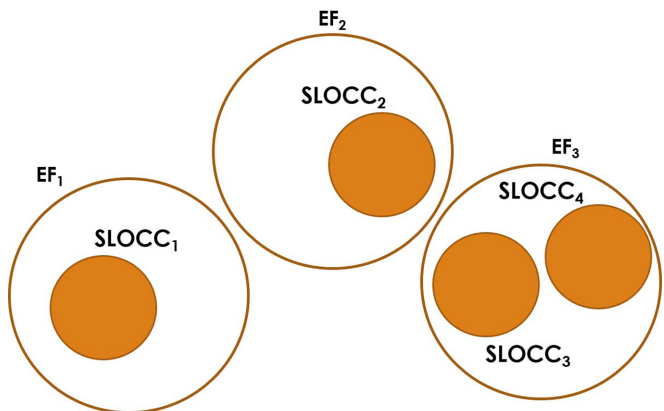


FIG. 5. Graphical representation of the relationship between EFs and SLOCC classes. Even if each EF does not contain only one SLOCC class, each SLOCC class is demonstrated to belong to only one EF [21].

III. QUANTUM CIRCUITS DESIGN WITH REINFORCEMENT LEARNING

A. Q-learning algorithm applied to quantum circuits design

In order to apply the Q-learning to our case, we implement the basic components of the algorithm as follows: (1) The *objectives*, which we will specifically search for, are the SLOCC classes (Table I) included in the nine EFs [Eq. (5)], hence, the representative states of the SLOCC classes of four qubits; (2) the *environment* is located in the four qubits state space; (3) the *actions* that the *agent* can perform consist in the application of quantum gates from a chosen set of gates; and (4) the *rewards* are encoded in a R-matrix whose entries corresponds to state-action pairs.

The training part consists in updating the Q-matrix whereas in the testing part the agent reaches the final objective state, producing the desired circuit. More specifically:

TABLE I. True SLOCC classes of four-qubit entangled states.

Family, SLOCC class	Conditions on coefficients
G_{abcd} , A1.1	$b = c = 0, ad \neq 0, a = \pm d$
G_{abcd} , A1.2	$b = c = 0, ad \neq 0, a \neq \pm d, a^2 + d^2 = 0$
G_{abcd} , A1.3	$b = c = 0, a^2 + d^2 \neq 0$
G_{abcd} , A2.1	$a = d, a \neq \pm b, b = +c, a^2 + b^2 = 0$
G_{abcd} , A2.2	$a = d, b = c, a \neq \pm b, a^2 + b^2 \neq 0$
G_{abcd} , A3.1	$a = d, a \neq \pm b, b = -c, a^2 + b^2 = 0$
G_{abcd} , A3.2	$a = d, b = -c, a \neq \pm b, a^2 + b^2 \neq 0$
G_{abcd} , A4.1	$a = d, \text{ either } a = \pm b \text{ or } \pm c, 2a^2 + b^2 + c^2 = 0$
G_{abcd} , A4.2	$a = d, \text{ either } a = \pm b \text{ or } \pm c, 2a^2 + b^2 + c^2 \neq 0$
G_{abcd} , A4.3	$a = d, a \neq \pm b, a \neq \pm c, 2a^2 + b^2 + c^2 = 0$
G_{abcd} , A4.4	$a = d, a \neq \pm b, a \neq \pm c, 2a^2 + b^2 + c^2 \neq 0$
G_{abcd} , A4.5	$a^2 + b^2 + c^2 + d^2 = 0$
G_{abcd} , A4.6	$a^2 + b^2 + c^2 + d^2 \neq 0$
L_{abc_2} , B1.1	$c = 0, a = b \neq 0$
L_{abc_2} , B1.2	$c = 0, a = -b \neq 0$
L_{abc_2} , B1.3	$c = 0, a \neq \pm b, a^2 + b^2 = 0$
L_{abc_2} , B1.4	$c = 0, a \neq \pm b, ab \neq 0, a^2 + b^2 \neq 0$
L_{abc_2} , B1.5	$c = 0, a \neq \pm b, ab = 0$
L_{abc_2} , B2.1	$abc \neq 0, a = b, a = \pm c$
L_{abc_2} , B2.2	$abc \neq 0, a = b, a \neq \pm c, a^2 + c^2 = 0$
L_{abc_2} , B2.3	$abc \neq 0, a = b, a \neq \pm c, a^2 + c^2 \neq 0$
L_{abc_2} , B3.1	$abc \neq 0, a = -b, a = \pm c$
L_{abc_2} , B3.2	$abc \neq 0, a = -b, a \neq \pm c, a^2 + c^2 = 0$
L_{abc_2} , B3.3	$abc \neq 0, a = -b, a \neq \pm c, a^2 + c^2 \neq 0$
L_{abc_2} , B4.1	$abc \neq 0, a \neq \pm b, a = \pm c, 3a^2 + b^2 = 0$
L_{abc_2} , B4.2	$abc \neq 0, a \neq \pm b, a = \pm c, 3a^2 + b^2 \neq 0$
L_{abc_2} , B4.3	$abc \neq 0, x \neq \pm y, a^2 + b^2 + 2c^2 = 0$
L_{abc_2} , B4.4	$abc \neq 0, x \neq \pm y, a^2 + b^2 + 2c^2 \neq 0$
L_{abc_2} , B5.1	$c \neq 0, a = b = 0$
L_{abc_2} , B5.2	$c \neq 0, a = 0, b = c$
L_{abc_2} , B5.3	$c \neq 0, a = 0, b \neq \pm c, b^2 + 2c^2 = 0$
$L_{a_2b_2}$, B5.4	$c \neq 0, a = 0, b \neq \pm c, b^2 + 2c^2 \neq 0$
$L_{a_2b_2}$, V1	$a = \pm b \neq 0$
$L_{a_2b_2}$, V2	$a \neq \pm b, ab \neq 0, a^2 + b^2 = 0$
$L_{a_2b_2}$, V3	$a \neq \pm b, ab \neq 0, a^2 + b^2 \neq 0$
$L_{a_2b_2}$, V4	$a \neq \pm b, ab = 0$
L_{ab_3} , R1.1	$a = b = 0$
L_{ab_3} , R1.2	$a = b \neq 0$
L_{ab_3} , R1.3	$a = -b \neq 0$
L_{ab_3} , R2.1	$a = 0, b \neq 0$
L_{ab_3} , R2.2	$a \neq 0, b = 0$
L_{ab_3} , R3.1	$a \neq \pm b, ab \neq 0, 3a^2 + b^2 \neq 0$
L_{ab_3} , R3.2	$a \neq \pm b, ab \neq 0, 3a^2 + b^2 = 0, b = +i\sqrt{3}$
L_{ab_3} , R3.2*	$a \neq \pm b, ab \neq 0, 3a^2 + b^2 = 0, b = -i\sqrt{3}$
L_{a_4} , La.1	$a = 0$
L_{a_4} , La.2	$a \neq 0$
$L_{a_20_{3\oplus\bar{1}}}$	$a \neq 0$
$L_{0_{5\oplus\bar{3}}}$	no conditions
$L_{0_{7\oplus\bar{1}}}$	no conditions

In this table we show the 49 true SLOCC classes identified in [22] among the nine families reported in Eqs.(5), with the conditions on the parameters $a, b, c,$ and d . Notice that the family $L_{0_{3\oplus\bar{1}}0_{3\oplus\bar{1}}}$ is not included because it is not characterized by four qubit entanglement since its representative state is a product state of the one-qubit state $|0\rangle$ and the three-qubit $|GHZ\rangle$ state [22].

a. Objectives. The algorithm can only focus on one target state at a time, thus we fix a single objective state, the representative of a SLOCC class in Table I. In order to suit the algorithm procedure, the representa-

tive state chosen is encoded as a list of its superposition terms, e.g., if the SLOCC class is A1.1 and the representative state reads $|\Psi\rangle_{A1.1} = \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle)$, then it is encoded as $\Psi_{A1.1} = \{0000, 1111\}$. This element is a

representation of the objective state that the algorithm can elaborate. In general, the state that we want to target can be written as

$$|\Psi\rangle = \sum_{j=1}^{16} \alpha_j |\psi\rangle_j, \quad (6)$$

where $|\psi\rangle_j$ are the sixteen basis states for the four-qubit state space, α_j are the superposition coefficients which can be either 0 or $\neq 0$. This state, for the purpose of the algorithm, can be encoded as

$$\Psi = \{\psi_1, \dots, \psi_n\} \quad (7)$$

with n being the number of terms of the objective quantum state $|\Psi\rangle$. Notice that in Eq. (7) only the basis states $|\psi_j\rangle$ with $\alpha_j \neq 0$ are listed. We specify that there is a clear distinction between $|\Psi\rangle$, the physical representation of the quantum state, and the abstract list-object Ψ , which represents it for the algorithm procedure. In fact, as we will see in the next sections, after the Q-learning procedure we need to make further manipulations on the output of the algorithm to obtain the physical objective state $|\Psi\rangle$ (see Appendix).

b. Environment. The environment of our algorithm is made up of quantum states of four qubits. We divide the Hilbert space into subsets characterized by states with a fixed number of terms in their superposition, i.e., single-term set, double-term set, etc. Moreover, in order to have a finite and discrete number of states to explore, as it is done for the *objectives*, we represent each state of the subsets as a list of its own superposition terms, without considering the states' coefficients. In fact, recording continuous coefficients would be an impossible task in terms of computational resources. However, to overcome the issues that can rise from this limitation, we will apply a post-processing procedure that allow us to tune the coefficients of the resulting state, in order to match the desired ones of the objective quantum state $|\Psi\rangle$ (see Appendix). If the representative state of the class has n terms, then the algorithm will explore the subsets with m terms, $m \leq n$. Notice that the dimension of the environment (the total dimension of the subsets involved) grows with the number of terms of the target state. This choice allows to explore a limited number of subsets.

Although this method limits the possibility to explore some of the four qubits entanglement classes, our algorithm proves to be an efficient way to design quantum protocols for a significant part of them.

c. Actions. The actions that the agent can perform consist in the application of single quantum gates (both single qubit and multiple qubits gates). The set of gates which the algorithm can apply is established before the training part. However, depending on our necessities we can update the set of gates before each search. For example, in order to reach specific classes, we added new gates with respect to the initial ones.

d. Rewards. The R-matrix entries correspond to the pairs *state-gate*, where the states are all the subsets of states with m terms, which compose the environment. The only entries that carry a value $\neq 0$ are those where the *gate* applied to the *state* gives, as a result, the target state.

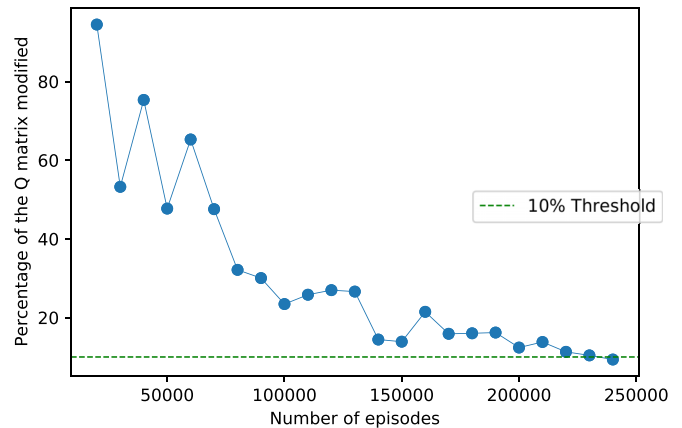


FIG. 6. Decreasing of the Q-matrix changing rate (CR) with the increasing number of episodes in the training part, for the search of the B1.1 SLOCC class (see Table I). We consider that the algorithm has successfully updated the Q-matrix when it reaches or passes the CR threshold.

B. Algorithmic procedure

When a target state $|\Psi\rangle$ is established, and then encoded in Ψ as a list-shaped object showed in Eq. (7), the algorithm first builds the R-matrix and initialize the Q-matrix as a zero matrix with the same shape as the R-matrix. We recall that the size of the R-matrix depends on the number of terms of the target state and on the number of gates that we decide to use.

a. Training part. During each episode of the training part, the algorithm is initialized in a random Ψ state, among the ones belonging to the subsets involved. Then it applies to the current state a random gate, taken from the gates set. It reads the resulting state and checks if the desired target state has been reached. By means of the Q-learning cost function [Eq. (1)], it updates the value of the Q-matrix entry, corresponding to the pair *current state-gate*. This procedure, consisting in a single episode of the Q-learning algorithm, is repeated for a fixed number of times, established before the training part. At the end of this run, we calculate what we can call the changing rate (CR). This quantity evaluates how much the Q-matrix values change, with respect to the values before the run. We can set a threshold under which the Q-matrix is considered substantially unchanged. After a sufficiently high number of episodes, if the algorithm reaches the desired threshold, we can consider that the Q-learning is at its convergence, and thus we can proceed to the testing part.

The number of repetitions needed to explore the whole environment depends on the dimension of this latter, which, as we mentioned, is linked to the number of terms of the searched state. In Fig. 6, we can see an example of this convergence procedure: the threshold for the CR is set at 10% and we can see how the CR of the Q-matrix decreases while number of episodes increases.

b. Testing part. In this part we check if the algorithm is able to find a suitable protocol to reach the objective state. We select an arbitrary initial state, e.g., $|0000\rangle$ encoded as $\{0000\}$. The algorithm checks the Q-matrix values in the cells that link the current state to the quantum gates, i.e., entries with coordinates $(\{0000\}, gate_i)$, with i scrolling all the

gates of the gate set. It then selects the action corresponding to the largest Q-value, and applies the gate to the current state. After reading the resulting state, it sets this latter as the new current state, and repeats the same procedure, until the final target state is reached. By recording the sequence of states and applied gates, it builds a quantum protocol, which starts from the arbitrarily chosen initial state and reaches to the objective one. Let us call $|\Psi\rangle_{\text{out}}$ the output state of the quantum protocol synthesized by the algorithm. The algorithm representation of the output state should be equal to the searched one Ψ : in other words, $|\Psi\rangle_{\text{out}}$ must have the same superposition terms showed in the Ψ list, i.e., the same superposition terms of $|\Psi\rangle$. This outcome proves that the training procedure was completed successfully, and thus the algorithm is able to produce the quantum circuit that generates the target state. As we already mentioned, in order to have an output state $|\Psi\rangle_{\text{out}}$ that matches also the coefficients of the quantum state $|\Psi\rangle$ we developed a post-processing procedure showed in Appendix.

IV. MACHINE LEARNING GENERATION OF ENTANGLED FOUR-QUBIT STATES





In this section, we apply the Q-learning algorithm to design proper quantum protocols to achieve the four qubits entangled classes. Using the classification in Table I [22], we present some of the quantum circuits generated by the algorithm to reach the representative states of those classes. We find out that not all these true SLOCC classes can be handled with the aid of our algorithm. Therefore we try to point out and explain the features of the circuits, which we manage to find for some of the classes, focusing on the specific gates that we introduced.

We start our search of quantum circuits with a simple set of gates, focusing firstly on the EF $L_{0_{3\oplus 1}0_{3\oplus 1}}$, i.e., the ninth EF. Although it is characterized by only three-partite entanglement, and thus it is not a true SLOCC class, it is a good starting point to test our algorithm, due to the possibility to validate our result with pre-existing literature concerning three qubits entangled states [17,31]. Moreover, this family representative state has only two terms meaning that the environment we have to build is extremely limited, thus, faster to explore. Therefore we take the representative state of the ninth family as a delightful example of the algorithm performance, and we present the results obtained with a simple set of gates, observing also its intrinsic limits.

Secondly, we show how this reduced gate set turns out to be not successful in reaching some of the other classes. Indeed, we take as an example the seventh and eighth EFs, since they coincide with true SLOCC classes [see Eqs. (5), Table I]. For these and other classes we are not able to produce a suitable circuit with the initial set of gates. Hence, to overcome this obstacle, we update it by adding the Toffoli gate: This allows us to reach some of the classes that were previously precluded.

Afterwards, we show that also this new gate-set is not yet sufficient, as it prevents to reach some of the classes whose representatives have an odd number of terms, as it will be elucidated afterwards. The solution that we found to this problem consists in the addition of the controlled-Hadamard (C-H) gate, to our gate set.

TABLE II. Universal set of quantum gates.

Name	Symbol	Matrix representation
Hadamard (H)		$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
Phase (S)		$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$
Control-not (C-NOT)		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$
Toffoli (C-C-NOT)		$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$

Universal set of quantum gates reported in [17].

By adding new gates to our set we manage to reach a large number of SLOCC classes in Table I, without altering the whole structure of the algorithm.

Finally, we summarize in Tables III–V which of the 49 true SLOCC classes we manage to reach with quantum protocols built by the algorithm or with the algorithm followed by the post-processing procedure in Appendix. Moreover, we suggest that some of the classes, which we did not manage to reach, could be approached with phase-dependent gates or with new subroutines that can be queued to our procedure.

A. First set of gates

As a universal set of quantum gates we introduce the set comprising the Hadamard gate, the phase gate, the controlled-not (C-NOT) gate, and the Toffoli gate (C-C-NOT) (see Table II). This is equivalent to the standard universal gates set which includes Hadamard, phase gate, C-NOT and $\pi/8$ gate [17,18]. However, as a first attempt we will use a reduced set of gates, made up of C-NOT, X gate (quantum NOT) and Hadamard. This choice is made aiming at reaching “simple classes” in the most direct and easiest way. Some of the classes, indeed, can be trivially obtained with this reduced set of gates, since approaching them with a universal set would introduce an unnecessary complication and would be more difficult in terms of computational resources. Thus we start with a simple set and, once we meet some obstacles with this initial toolbox, we add new gates. With this approach it can be argued that we end up with a set which includes a universal set but has some redundancy, meaning that not all the gates are independent. However, by adding redundancy we gain in efficiency; indeed, a richer toolbox improves the performances of the learning part in terms of computational time.

Furthermore, by including unconventional gates in the toolbox (such as controlled-Hadamard) we help the algorithm in finding optimal circuits characterized by a smaller number of gates. Moreover, handling different entanglement classes with different sets of gates, growing in complexity, allows us

TABLE III. Feasibility categorization.

Family, SLOCC class	Conditions on parameters	Feasibility
G_{abcd} , A1.1	$b = c = 0, ad \neq 0, a = \pm d$	●
G_{abcd} , A1.2	$b = c = 0, ad \neq 0, a \neq \pm d, a^2 + d^2 = 0$	●
G_{abcd} , A2.1	$a = d, a \neq \pm b, b = +c, a^2 + b^2 = 0$	●
G_{abcd} , A3.1	$a = d, a \neq \pm b, b = -c, a^2 + b^2 = 0$	●
L_{abc_2} , B1.1	$c = 0, a = b \neq 0$	●
L_{abc_2} , B1.2	$c = 0, a = -b \neq 0$	●
L_{abc_2} , B1.3	$c = 0, a \neq \pm b, a^2 + b^2 = 0$	●
L_{abc_2} , B1.5	$c = 0, a \neq \pm b, ab = 0$	●
L_{abc_2} , B2.1	$abc \neq 0, a = b, a = \pm c$	●
L_{abc_2} , B2.2	$abc \neq 0, a = b, a \neq \pm c, a^2 + c^2 = 0$	●
L_{abc_2} , B3.1	$abc \neq 0, a = -b, a = \pm c$	●
L_{abc_2} , B3.2	$abc \neq 0, a = -b, a \neq \pm c, a^2 + c^2 = 0$	●
L_{abc_2} , B5.1	$c \neq 0, a = b = 0$	●
L_{abc_2} , B5.2	$c \neq 0, a = 0, b = c = 1$	●
L_{abc_2} , B5.3	$c \neq 0, a = 0, b \neq \pm c, b^2 + 2c^2 = 0$	●

In this table, we report the feasibility categorization of our algorithm and post-processing procedure for the SLOCC classes belonging to the EFs G_{abcd} and L_{abc_2} .

to better observe their entanglement properties. For example, some of the entanglement classes explicitly need gates which can act on three qubits (see Sec. IV B for the introduction of the Toffoli gate), while some other classes do not require this kind of gates. In other words, adding gates step by step helps us to point out their key role in reaching specific SLOCC classes.

As anticipated, we firstly choose to analyze the ninth EF. Because its environment is extremely limited, the time needed to check if the algorithm succeeds is relatively short. In fact, since the representative of the class reads

$$|\Psi\rangle_{L_{0_3 \otimes 1^0 3 \otimes 1}} = |0000\rangle + |0111\rangle, \tag{8}$$

its algorithm representation reads $\{0000, 0111\}$, thus the only rewarding matrices, and quality matrices, which we need to build are those that connect *actions*, i.e., application of gates,

to single-term states (ST) and double-term states (DT). The ST and DT lists reads

$$ST = \begin{Bmatrix} 0000 \\ 0001 \\ 0010 \\ \vdots \end{Bmatrix} \quad DT = \begin{Bmatrix} \{0000, 0001\} \\ \{0000, 0010\} \\ \{0010, 0011\} \\ \vdots \end{Bmatrix}.$$

From now on, we will refer to the representative states of the nine families as $|\Psi\rangle_J$, with $J = 1, \dots, 9$, and the representative states of the SLOCC classes with the notation used in Table I [22]. Thus $|\Psi\rangle_{L_{0_3 \otimes 1^0 3 \otimes 1}} = |\Psi\rangle_9$.

In Fig. 7, we can see the R-matrix of the ninth EF of Eq. (8) for the DT states, carrying rewards only in correspondence of some *state-action* pairs, highlighted in green. After training the algorithm, the Q-matrices are built: In Fig. 8, we can see the insight of the Q-matrix for the ST subset. The pattern of

TABLE IV. Feasibility categorization.

Family, SLOCC class	Conditions on parameters	Feasibility
$L_{a_2 b_2}$, V1	$a = \pm b \neq 0$	●
$L_{a_2 b_2}$, V2	$a \neq \pm b, ab \neq 0, a^2 + b^2 = 0$	●
$L_{a_2 b_2}$, V4	$a \neq \pm b, ab = 0$	●
L_{ab_3} , R1.1	$a = b = 0$	●
L_{ab_3} , R1.2	$a = b \neq 0$	●
L_{ab_3} , R1.3	$a = -b \neq 0$	●
L_{ab_3} , R2.1	$a = 0, b \neq 0$	●
L_{ab_3} , R2.2	$a \neq 0, b = 0$	●
L_{ab_3} , R3.2	$a \neq \pm b, ab \neq 0, 3a^2 + b^2 = 0$	●
L_{ab_3} , R3.2*	$a \neq \pm b, ab \neq 0, 3a^2 + b^2 = 0$	●
L_{a_4} , La.1	$a = 0$	●
L_{a_4} , La.2	$a \neq 0$	●

Here we show the feasibility for the EFs $L_{a_2 b_2}$, L_{ab_3} , and L_{a_4} . We can see that, due to the complex coefficients of the EF L_{ab_3} in Eqs. (5), the SLOCC classes related to this EF are not approachable with our post-processing procedure.

TABLE V. Feasibility categorization.

Family, SLOCC class	Conditions on parameters	Feasibility
$L_{a_2 0_{3\oplus\bar{1}}}$	$a \neq 0$	●
$L_{0_{5\oplus\bar{3}}}$	no constraints	●
$L_{0_{7\oplus\bar{1}}}$	no constraints	●

The last three SLOCC classes that we report are among those showed as examples in this paper, $|\Psi\rangle_6$, $|\Psi\rangle_7$, and $|\Psi\rangle_8$.

weighted rewards, i.e., the quality values $Q(s, a)$, is clearly visible. During the testing part, as explained in Sec. III B, we choose a starting quantum state, set as the *current state*, that will be the initialized state in our quantum circuit. The algorithm checks the Q-matrix values in the cells that link the current state representation to the quantum gates, i.e., if the chosen quantum state is $|\psi_0\rangle = |0000\rangle$, it checks the entries with coordinates $(\{0000\}, gate_i)$, with i scrolling all the gates of the gate set. It then selects the largest Q-value in that row and applies the correspondent gate to the current state. It registers the resulting state and sets this latter as the new current state, and repeats the same procedure, until the final target state is reached. In Fig. 9, the result of the testing part is displayed in form of a quantum circuit, as the sequence of gates selected as the *best valued* once. The sequential appli-

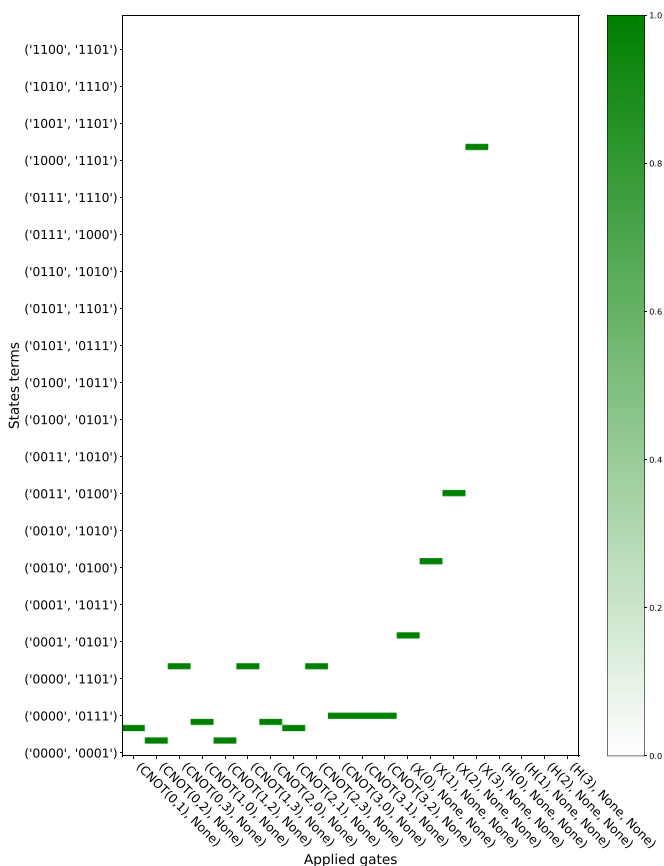


FIG. 7. R-matrix for the DT states for the EF $|\Psi\rangle_9$, i.e., $L_{0_{3\oplus\bar{1}} 0_{3\oplus\bar{1}}}$. The green cells represent the rewards, and they are present only for the pairs *state-action* that directly link to the objective class.

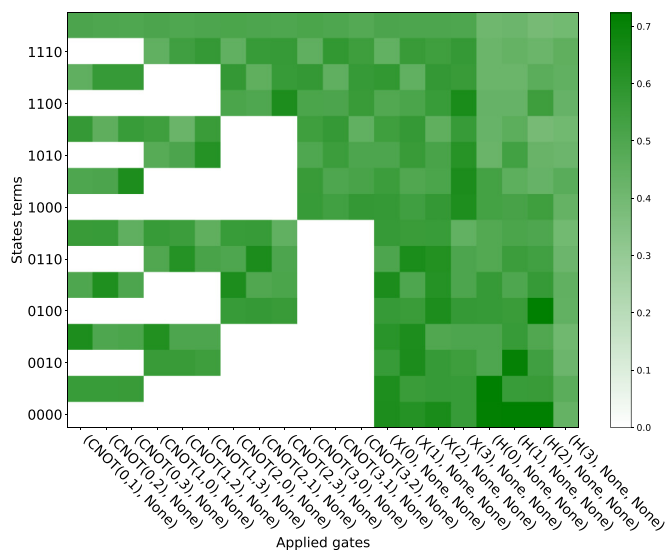


FIG. 8. Q-matrix of the ninth EF, built for the ST states. The cells represent the pairs *state-action* and the color-grade indicates the Q-value itself. While the R-matrix include just the single-gate rewards, this matrix contains a pattern of rewards, which help the algorithm to choose the best gate to apply when placed in a certain state.

cation of the gates produces the output state $|\Psi\rangle_{out}$ which, in terms of its algorithm representation Ψ_{out} , meets exactly the desired representative state $|\Psi\rangle_9 \rightarrow \Psi_9 = \{0000, 0111\}$. We will introduce in the next section an intuitive way to visualize states and gates connections. Based on this first successful result we may start dealing with more time-demanding classes.

In Secs. IV B and IV C, we explain why we decide to add new gates to the reduced toolbox, by elucidating what kind of difficulties we encounter trying to reach some of the entanglement classes.

B. Adding the Toffoli gate

Two of the entanglement classes which the algorithm struggled to handle at this stage are the seventh and the eight:

$$|\Psi\rangle_7 = |0000\rangle + |0101\rangle + |1000\rangle + |1110\rangle, \quad (9)$$

$$|\Psi\rangle_8 = |0000\rangle + |1011\rangle + |1101\rangle + |1110\rangle. \quad (10)$$

For an entangled state with four terms, like $|\Psi\rangle_7$ and $|\Psi\rangle_8$, we have to explore all the subsets including states with $n \leq 4$ terms, i.e., ST subset, DT subset, three-term subset (TT),

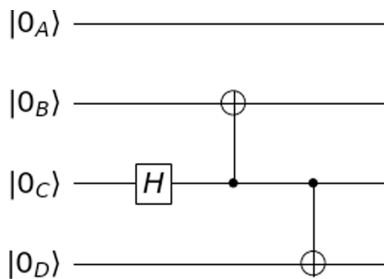


FIG. 9. Quantum circuit for the $|\Psi\rangle_9$ family resulting from the reinforcement learning algorithm.

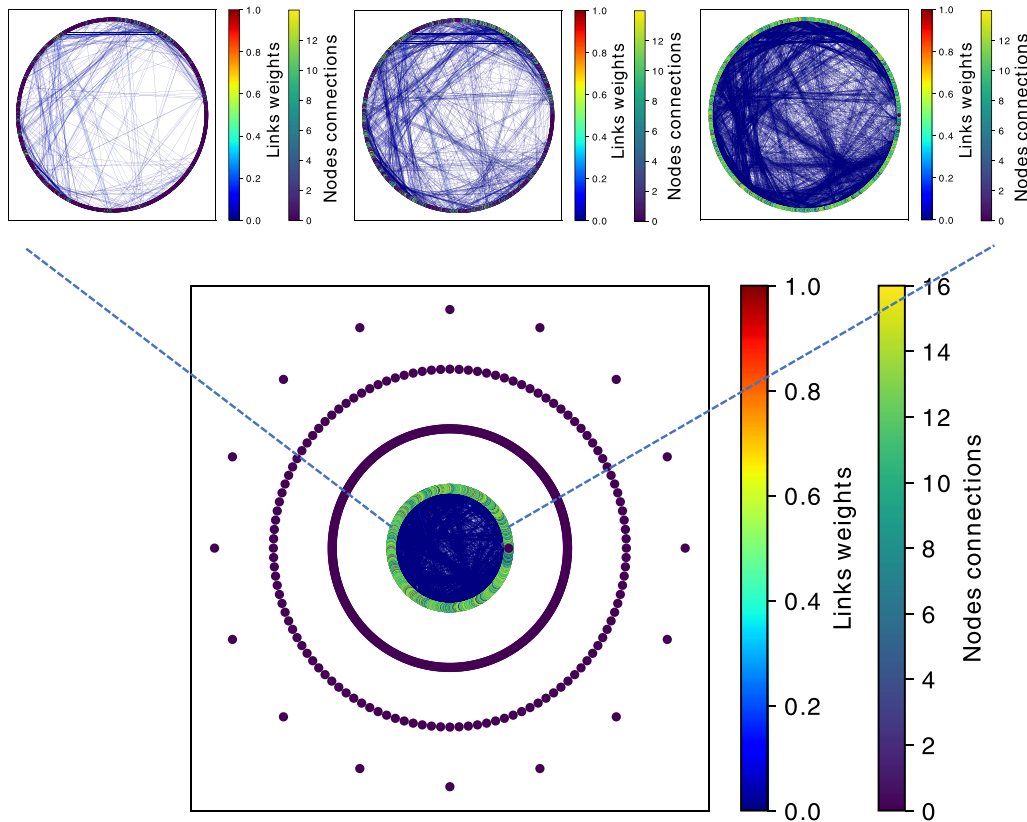


FIG. 10. State-link graph (SLG) encoding the information of the Q-matrix for the $|\Psi\rangle_7$ SLOCC class. Each circular shell represents a subset of states, from the outer one to the inner one: Single-term (ST), double-term (DT), three-term (TT) and four-term (FT). The nodes represent the states, while the rewarded single gate transformations are represented by the links between nodes. The color grade of the nodes refers to the number of connections that each state has with other states, belonging to the same shell or to different shells. Whereas the color of the edges indicates the weighted reward associated to that gate application. As we can see the rewards only spread inside the FT shell. On the top are showed the insights of the inner shell related to three steps (from left to right) of the progressive training procedure. We can see how the FT states are progressively connected by rewarded gates applications.

and four-term subset (FT). During the training part, even if the algorithm manages to converge in terms of the Q-matrix modifications, it shows that the rewards do not propagate outside the FT subset. Indeed, we noticed that, at the end of the training, only the Q-matrix related to the FT states reports weighted rewards, while the other three Q-matrices (related to TT, DT, and ST states) remain unchanged. This means that, with the set of gates provided, there are no rewarded links between the FT subset and the TT, DT, and ST subsets.

This affects our capability to reach the desired entangled state from an arbitrary quantum state in our environment. Indeed, if the agent is placed in one of the TT, DT, or ST states, it will have no clue on which is the best action to take in order to reach the desired state in the four-term subset, due to the fact that there are no rewards spread in that part of the environment where it is located. As a consequence, during the testing part, if the agent is placed in one of the ST, DT, or TT subsets, it will only take random actions, because all the state-action pairs have the same quality weight (equal to zero).

We can visualize the connections between quantum states, encoded in the Q-matrix, with the state-link graph (SLG) shown in Fig. 10. The nodes of the graph represent the

environment states and the links which connect two states correspond to the rewarded application of a single gate. Different concentric shells correspond to different subsets, i.e., to subsets which states have different number of terms. The innermost shell is the one related to the four-term states: it is the shell where the representative state of the class is located. The other shells, from the outer one to the inner one, are the ST, DT, and TT states. The color grade of the nodes refers to the number of connections that each state has with other states and the colors of the links correspond to the Q-values. We can see that the nodes of the outer shells have no rewarded connections, while the four-term shell has links between its states. This behavior indicates that the algorithm could not spread the reward pattern in the Q-matrix outside a certain region of the environment. If we want our algorithm to find the optimal path starting from an arbitrary initial state, we have to provide shortcuts that can connect different shells with a rewarded link. Notice that this obstacle cannot be traced back to the number of terms in the superposition of the objective state, because for other classes with four terms the algorithm worked efficiently. As an example we can look at the entanglement family G_{abcd} , in particular, the SLOCC class A4.1 in Table I, where the parameters are set as $a = d$,

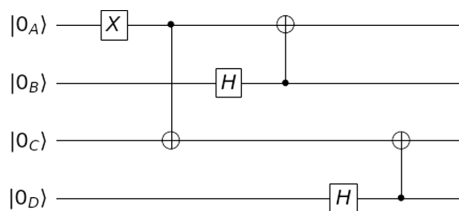


FIG. 11. Quantum circuit for representative state $|\Psi\rangle_{A4.1}(a = 0)$.

$a = \pm b, 2a^2 + b^2 + c^2 = 0$, with $a = 0$:

$$|\Psi\rangle_{A4.1}(a = 0) = |0101\rangle + |1010\rangle + |0110\rangle + |1001\rangle$$

↓ algorithm representation

$$\Psi_{A4.1}(a = 0) = \{0101, 1010, 0110, 1001\}. \quad (11)$$

For this class, the rewards spread over all the states subsets and the algorithm is able to find the circuit showed in Fig. 11, starting from the state $|\psi_0\rangle = |0000\rangle$. The reason for this behavior lies on the particular terms of the superposition characterizing the two classes $|\Psi\rangle_8$ and $|\Psi\rangle_7$: we can see that both have one or more terms with three qubits in the state $|1\rangle$. In fact $|\Psi\rangle_8$ is in a superposition of $|1011\rangle, |1101\rangle$, and $|1110\rangle$, and $|\Psi\rangle_7$ includes $|1110\rangle$. To account for this feature it becomes necessary to introduce the Toffoli gate (C-C-NOT). It performs a quantum not on a target qubit only if the two control qubits are in the state $|1\rangle$ at the same time. Indeed, acting on three qubits the Toffoli gate generates this type of superposition without creating or canceling other terms in the state. For our purpose, the Toffoli gate allows the algorithm to reach superpositions between base elements which include terms with three qubits in the state $|1\rangle$. Indeed, after the introduction of C-C-NOT in the set of gates, the algorithm manages to spread the reward and finds paths that allow to reach $|\Psi\rangle_7$ starting from an arbitrary quantum state. In Fig. 12, we can see the quantum circuit for the seventh EF, with $|\psi_0\rangle = |0000\rangle$. We can also observe the Q-matrices pattern of rewards, once the Toffoli is introduced: in fact, the state-link graph (SLG) appears completely filled with rewarded links that connect different shells Fig. 13. With the addition of the Toffoli gate we almost reached a universal set of gates, which comprehends: Hadamard gate, Z gate, C-NOT, and Toffoli gate [17]. However, due to the fact that we are not keeping track of the amplitudes, we do not need to add the phase gate Z.

C. Adding the controlled-Hadamard gate

Proceeding with the analysis of the SLOCC classes we find that, at this stage, the algorithm is unable to reach some

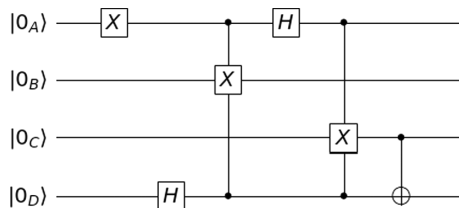


FIG. 12. Quantum circuit built for $|\Psi\rangle_7$ with the aid of the Toffoli gate.

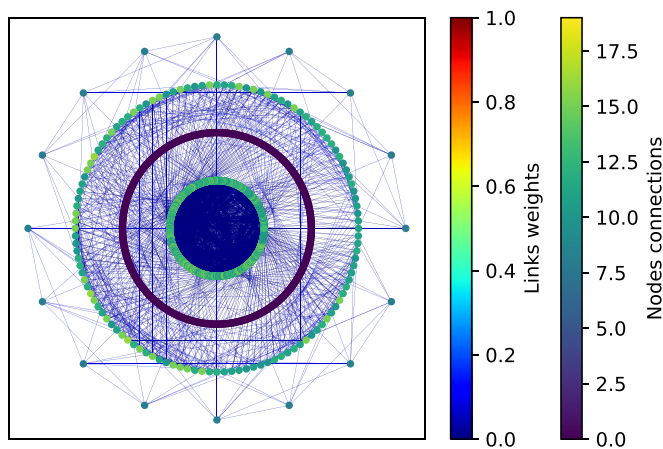


FIG. 13. State-link graph (SLG) for the objective state $|\Psi\rangle_7$ after the introduction of the Toffoli gate. In this graph, the four shells are completely connected by weighted rewards, as opposed to the situation in Fig. 10.

states with an odd number of terms in their superposition. In particular, states which representative has three terms, such as the representatives of classes B1.1, B1.2, and B5.1. Again we decide to add a gate to our toolbox, in order to reach these SLOCC classes. The best candidate that can help us is the Hadamard gate, because it introduces superposition. Indeed, its action reads

$$H|0\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad H|1\rangle \rightarrow \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \quad (12)$$

In terms of a quantum optics device, by taking the polarization as our degree of freedom and considering linearly polarized photons, the Hadamard corresponds to a polarization rotator which performs a rotation of $\pi/4$, i.e., a half-wave plate. Indeed, horizontal and vertical polarization can be encoded, respectively, as $|0\rangle$ and $|1\rangle$ of the computational basis. A qubit subject to the action of a Hadamard gate is analogous to a photon passing through a half-wave plate: it is projected onto the basis $\{|+\rangle = 1/\sqrt{2}(|0\rangle + |1\rangle), |-\rangle = 1/\sqrt{2}(|0\rangle - |1\rangle)\}$ [33].

Applying Hadamard to a single-term state we obtain a double-term state. Of course, applying it to a double-term state we may obtain a three-term state. However, when we look at what happens when a Hadamard is applied to a double-term state, *in an homogeneous superposition*, we can see that there are just a few possible outcomes.

We have two different cases, depending on the entanglement. If the state is entangled and it is in an homogeneous superposition, the action of the Hadamard gate can only lead to one type of result, e.g., on one of the Bell's states it reads

$$H(A)(|00\rangle + |11\rangle) \rightarrow |00\rangle + |10\rangle + |01\rangle - |11\rangle, \quad (13)$$

where $H(A)$ is the Hadamard acting on the first qubit. A similar superposition of four terms can be obtained from the action of one Hadamard gate on the other three Bell states. The other possibility is that the two-qubit state, on which the Hadamard acts is not entangled, then the result can be summarized in the following way.

(i) Action on the factorizable qubit

$$\begin{aligned} H(A)(|01\rangle + |00\rangle) &\rightarrow H(A)|0\rangle_A(|1\rangle_B + |0\rangle_B) \\ &\rightarrow |01\rangle + |11\rangle + |00\rangle + |01\rangle. \end{aligned}$$

(ii) Action on the nonfactorizable qubit

$$\begin{aligned} H(B)(|01\rangle + |00\rangle) &\rightarrow |0\rangle_A H(B)(|1\rangle_B + |0\rangle_B) \\ &\rightarrow |00\rangle. \end{aligned}$$

This means that the Hadamard gate, acting on the two qubits states in homogeneous superpositions, can only double the number of terms (from two to four terms) or halves it, returning to a single-term state. This discussion can be generalized to the three and four qubit case. Indeed, as the Hadamard gate is a single-qubit gate, if we want to obtain a recombination of terms, i.e., to sum two or more terms, we need superpositions of terms, which differ from each other for a single qubit and the Hadamard gate should act on that specific qubit:

$$H(A)(|01\rangle + |11\rangle) \rightarrow |01\rangle + |11\rangle + |01\rangle - |11\rangle \rightarrow |01\rangle. \quad (14)$$

If we apply the Hadamard gate to a two-term state of four or three qubits with terms which differ for more than one qubit, we will obtain always four terms as a result, because the sum of the terms would not be possible, keeping in mind that the Hadamard acts on only one qubit at a time. This observation leads to the following results regarding the four qubits states, analogues to those listed above for the two-qubit states.

(i*) The two terms differ from each other for more than one qubit, and the Hadamard gate acts on one of them

$$\begin{aligned} H(B)(|0100\rangle - |0010\rangle) &\rightarrow |0000\rangle - |0100\rangle \\ &\quad - |0010\rangle + |0110\rangle. \end{aligned}$$

We obtain a four-term superposition from a two-term one.

(ii*) The two terms differ one another for more than one qubit and the Hadamard gate acts on the factorizable ones (analogue to the case *i*)

$$\begin{aligned} H(A)(|0100\rangle + |0010\rangle) &\rightarrow |0100\rangle + |1100\rangle \\ &\quad + |0010\rangle + |1010\rangle. \end{aligned}$$

We obtain a four-term superposition from a two-term one.

(iii*) The two terms differ for a single qubit and the Hadamard gate acts on the factorized ones (this case is not different from the previous one *ii**, due to the fact that the Hadamard gate can only change one qubit at a time)

$$\begin{aligned} H(A)(|0100\rangle + |0000\rangle) &\rightarrow |0100\rangle + |1100\rangle \\ &\quad + |0000\rangle + |1000\rangle. \end{aligned}$$

We obtain a four-term superposition from a two-term one.

If the two terms differ by only one qubit, which corresponds to the qubit targeted by the Hadamard gate, then the result can be trivially referred to the two-qubit case (ii), without loss of generality.

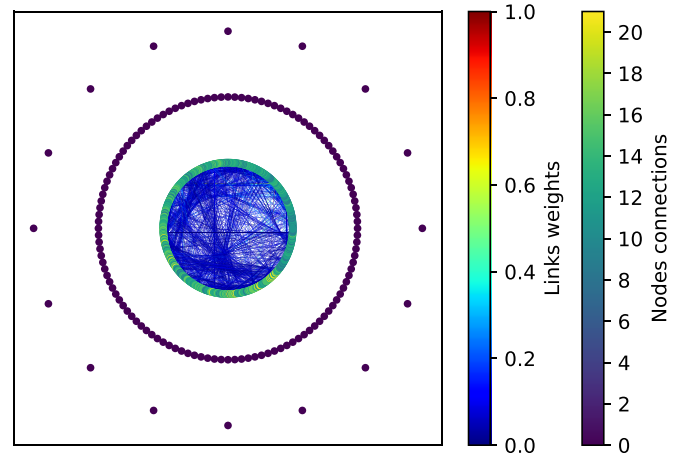


FIG. 14. The state-link graph (SLG) for the objective state $|\Psi\rangle_{B1.1}$ without the C-H gate added, showing that the shells related to ST states (outer shell) and DT (next-to-outer shell) states are not connected by rewarded single gates applications.

(iv*) The two terms differ for a single qubit and the Hadamard gate acts on it

$$\begin{aligned} H(D)(|0000\rangle + |0001\rangle) &\rightarrow |0000\rangle + |0001\rangle \\ &\quad + |0000\rangle - |0001\rangle \\ &\rightarrow |0000\rangle. \end{aligned}$$

We obtain a state which is no more in a superposition, a single-term state from a two-term one.

In polarization terms, we are performing a rotation from the diagonal basis $|+\rangle_D = 1/\sqrt{2}(|0\rangle_D + |1\rangle_D)$, $|-\rangle_D = 1/\sqrt{2}(|0\rangle_D - |1\rangle_D)$ to the basis $|0\rangle_D, |1\rangle_D$. Thus the Hadamard gate *cannot provide three terms in a superposition*.

The discussion becomes different if we consider states of four terms and we want to reach five terms. In that case the Hadamard gate, together with the action of NOT or C-NOT gates, can recombine terms and a state with five terms can be built from a four-term state. Since some of the entanglement classes that we want to reach have representative states with three terms, we choose to add a new gate to the previous set of gates, namely, the Controlled-Hadamard (C-H) gate. To see the effectiveness of this addition, we take as an example the SLOCC class B1.1, whose representative state is $|\Psi\rangle_{B1.1} = |0000\rangle + |1111\rangle + |0110\rangle$ [22]. In Fig. 14, we can see the SLG related to this entanglement class without the addition of the C-H gate. We notice that the TT states are isolated from the others, as it happened before with $|\Psi\rangle_7$. With the C-H included in the toolbox we can overcome this obstacle and we are able to generate states with three terms in their superposition. Without loss of generality, we can make an example of the action of a C-H on a simple two-qubit state, omitting normalization coefficients:

$$C-H(B, A)(|01\rangle_{AB} + |10\rangle_{AB}) \rightarrow |01\rangle + |11\rangle + |10\rangle. \quad (15)$$

In this example, the C-H gate has qubit B as the control qubit and qubit A as the target. Due to the nonsymmetric action of this gate we can reach states with an odd number of terms with a shortcut which connects even-term shells and odd-term

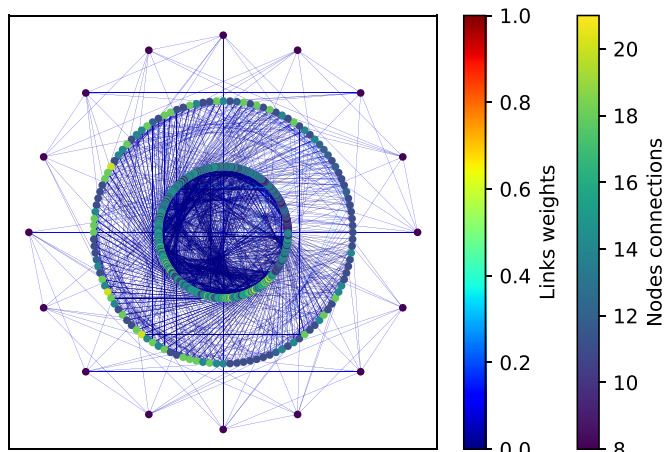


FIG. 15. The state-link graph (SLG) for the objective state $|\Psi\rangle_{B1.1}$ with the C-H gate added showing how the shells, corresponding to all number of terms, link with one another.

shells in the SLG. In Fig. 15, we can see how the graph related to $|\Psi\rangle_{B1.1}$ is now connected by edges that allow to reach the objective state. We report the quantum circuit to generate $|\Psi\rangle_{B1.1}$ in Fig. 16, and the corresponding optimal path in a graph form in Fig. 17.

D. Complete and incomplete exploration of the entanglement classes

We summarize in Tables III–V the subgroup of the 49 classes, showed in Table I, which we managed to approach with our reinforcement learning algorithm. We recall that these SLOCC classes are identified from the original nine entanglement families in Eqs. (5), with constraints on the complex parameters $a, b, c,$ and d [22]. The colored dots in the *Feasibility* column have the following meaning:

● → class that we are able to reach with the set of gates $\{X, H, C\text{-NOT}, C\text{-C-NOT}, C\text{-H}\}$, without the post-processing procedure,

● → class that can be reached with the previously mentioned set of gates and the post-processing procedure described in Appendix,

● → class that requires post-processing procedure shown in Appendix and phase gates addition.

For the SLOCC classes marked with a dark green dot (●), we are able to produce the quantum circuits generating their representative state exploiting only the Q-learning procedure. We showed some of them in the previous sections. These

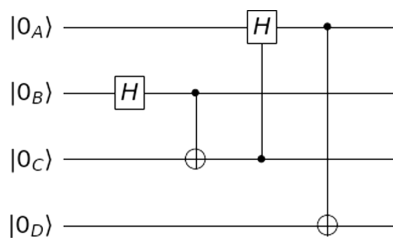


FIG. 16. Quantum circuit obtained for the SLOCC class B1.1 (Table I) generated with the reinforcement learning algorithm.

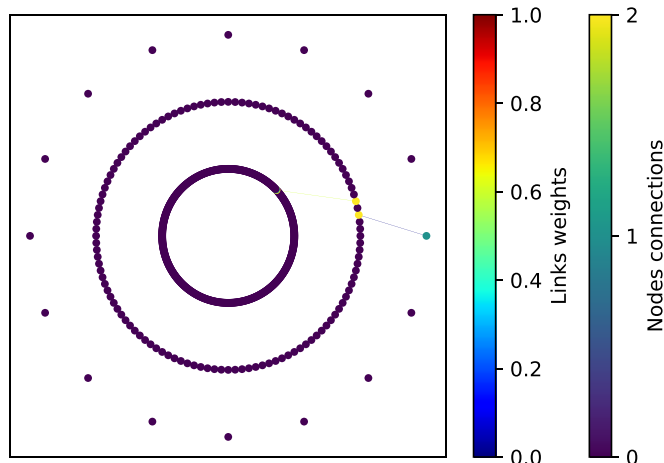


FIG. 17. Quantum circuit for the class B1.1 in the SLG representation. These links represent connections created by the single gates of the protocol in Fig. 16. The starting state on the right is $\Psi_0 = \{0000\}$.

classes have representative states that are homogeneous superpositions with real coefficients, i.e., their terms have all the same real coefficients. For these classes, we managed to find suitable protocols, reported in Sec. IV E, which can reproduce their representative states. We consider these results optimal given the settings of the algorithm.

The classes with a blue mark (●) required the post-processing procedure, described in Appendix. Indeed, even though the circuit produced by the Q-learning procedure has an output state with the same terms of the representative one, we need to tune the coefficients of this output state, to match those of the desired state.

The classes with a yellow mark (●) have representative states in which one or more terms have imaginary coefficients. Hence, our algorithm is able to reach states with the right terms of the superposition but our post processing procedure is not able to match the coefficients in the end, as it is designed only to tune real coefficients. These states can be reached with a proper addition of the phase gates (S), the control-phase gates (C-S) or the $\pi/8$ gates. By means of them we can rearrange the phases of the coefficients accordingly, in order to achieve the desired representative state. In any case, our algorithm can be used as a fruitful starting point.

There are some classes in Table I that are not included in this summary. This is because they are not fully classified [22], meaning that they could either be true SLOCC classes or not. Let us recall that this feasibility categorization is based on the capabilities of our algorithm and it is intrinsically linked to its working principles. Thus, among the classes listed in this section, we consider *completely explored* the classes marked in dark green and blue, while we consider the others as partially explored, which could be finalized along the lines mentioned here or by further developments.

We divide the SLOCC classes into three tables. The first one, Table III, refers to the SLOCC classes that derive from the first two EFs in Eqs. (5). We notice that the four-qubit GHZ state is included in the first EF, the representative state

TABLE VI. Quantum protocols.

SLOCC class	Quantum protocol
Representative state	
A1.1 $ 0000\rangle + 1111\rangle$	
B1.1 $ 0000\rangle + 1111\rangle + 0110\rangle$	
B1.2 $ 0011\rangle + 1100\rangle + 0110\rangle$	
B2.1 $ 0000\rangle + 1111\rangle + 0101\rangle + 1010\rangle + 0110\rangle$	
B3.1 $ 0011\rangle + 1100\rangle + 0101\rangle + 1010\rangle + 0110\rangle$	

Quantum protocols designed with the algorithm for some of the SLOCC classes of four-qubit entangled states.

of class A1.1. being

$$|\text{GHZ}\rangle_4 = \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle). \quad (16)$$

In Table IV, the SLOCC classes that derive from the third, fourth, and fifth families in Eqs. (5) are reported.

In Table V, are reported the SLOCC classes belonging to the remaining families in Eqs. (5), the sixth, the seventh and the eighth families, since the ninth family does not contain four qubits entanglement.

E. Quantum circuits for SLOCC classes of four qubits

This subsection summarizes the main results of this work. We report in Tables VI and VII the quantum circuits that we find with our reinforcement learning algorithm for some of the classes listed in Tables III–V, marked with a blue (●) and a dark green dot (●). We can see that, in some cases the algorithm uses the Toffoli gate and the C-H gate extensively: this is due to the optimization of the circuitual length.

TABLE VII. Quantum protocols.

SLOCC class	Quantum protocol
Representative state	
B5.1 $ 0101\rangle + 1010\rangle + 0110\rangle$	
V4 $ 0000\rangle + 1111\rangle + 0110\rangle + 0011\rangle$	
R1.1 $ 0001\rangle + 0010\rangle + 0111\rangle + 1011\rangle$	
La.1 $ 0001\rangle + 0110\rangle + 1011\rangle$	
$La_2 0_{3\oplus\bar{1}}$ $ 0000\rangle + 1111\rangle + 0011\rangle + 0101\rangle + 0110\rangle$	
$L_{0_{5\oplus\bar{3}}}$ $ 0000\rangle + 0101\rangle + 1000\rangle + 1110\rangle$	
$L_{0_{7\oplus\bar{1}}}$ $ 0000\rangle + 1011\rangle + 1101\rangle + 1110\rangle$	

Quantum protocols for some of the SLOCC classes of four-qubit entangled states.

V. CONCLUSIONS

We exploited the potentiality of the off-policy Q-learning algorithm that provided us a tool for state preparation in quantum computing. We focused on the interplay between the chosen sets of gates, the encoding method and visualization in SLGs, and the entanglement SLOCC classes. The idea to use a single gate reward carried to highlight the role of some specific single gates in reaching states with some qualitative entanglement features. We pursued the idea of giving an oper-

ative indication on the complexity of qubits state preparation, in terms of circuit composition.

We have shown that with our implementation of the Q-learning algorithm, we manage to successfully build quantum protocols able to generate representative states for some of the 49 true SLOCC classes of the four-qubit entanglement states. In particular, we are able to reach at least one true SLOCC class for each of the nine entanglement families. Further, we observe that many of the other SLOCC classes can be approached by adding other quantum gates to the set, and modifying accordingly the algorithm in order to use them. Therefore this machine learning algorithm is useful in reaching a large number of four-qubit entangled states, and could be employed to better understand their properties and to devise new procedures to construct them in a real experiment. Moreover, our method in principle allows to deal with n qubit states, four-party quantum states with multilevel systems, i.e., we could in future works extend it from qubits to qutrits or higher, thereby it can in principle produce further new results in unknown territory.

Furthermore, we can discover new connections between specific entanglement features and the role of certain quantum gates. In this sense, thanks to its simplicity and intuitiveness, the Q-learning algorithm turns out to be widely profitable for these kind of tasks. Something similar to this was discovered with the Melvin algorithm [6,7] for a low number multidimensional (a few qutrits, etc.) quantum states. It is conceivable that those multidimensional quantum states can be addressed with the tools developed in this work.

Due to our limited computational resources and the intricacies of some of the entangled states addressed, we have not completed the full generation of all the 49 classes of entangled four qubit states. Thus a possible next step to take to reach the representatives of the remaining classes would be to make the algorithm capable of handling states in a nonhomogeneous superposition, with complex coefficients, without the aid of a post processing method. In this way, the number of SLOCC classes for which we could be able to provide protocols will further increase. Therefore, even if the Q-learning in its current form is not suitable for the very complete exploration of the nine entanglement families, it provides reasonable clues as to how to attempt to handle the unsolved cases, if further capabilities are included in the resource toolbox of the reinforcement Q-learning algorithm.

We have devised a graphical tool called the state-link graph (SLG) to represent the construction of the Q-matrix for a given objective state belonging to one of the entanglement classes. See examples in Figs. 10, 13, etc. These graphs are very useful to detect whether the learning algorithm is exploring the set of multiple terms needed to reconstruct the objective state. This way, when it is detected that some of the shells in SLG are not connected, it is an indication that our gate-set chosen is not rich enough so as to build the given state. Then, it means that it is the moment to enlarge the quantum gate-set. This is precisely the process that we have followed to synthesize the quantum circuits found in Tables VI and VII.

Some of the results obtained for the synthesis of four-qubit states with remarkable entanglement properties, such as those in Tables VI and VII, may be useful to investigate statements about the local and realistic properties of our universe with ex-

perimental means as was originally proposed with the Melvin algorithm [6,7]. In our Q-learning algorithm, we do not need to know about the concept of Schmidt coefficients as the Melvin algorithm does. The reinforcement machine learning algorithm does not rely on previous knowledge nor on often flawed intuition.

By construction, the quantum circuits found with our reinforcement learning algorithm are optimal with respect to the quantum gate-set chosen. This is guaranteed by the convergence of the training part and the subsequent construction of the quantum circuits in the testing part relying on optimal state-action pairs found for the Q-matrix. This result is useful for the automated quantum circuit synthesis (QCS) where optimal implementations of quantum algorithms are designed from quantum logic gates belonging to known universal sets [34–38]. These are the type of automated tasks needed to construct quantum compilers. Machine learning methods to synthesize optimal circuits for continuous variable quantum computation have been proposed with photonics architectures [39,40].

It is worth noticing that our reinforcement learning algorithm is not scalable as the number of entangled qubits increases since the number of multiple terms needed to construct the objective states and the environment, grows exponentially with the number of entangled qubits. Thus, although we can boost the task of automatically constructing the quantum circuits for many of the entanglement classes of four qubit states, in the end we will also face the wall of the exponential complexity of quantum entangled states with an arbitrary number of qubits. Nevertheless, the quantum circuits synthesized with machine algorithms with reinforcement learning can serve as a benchmark for more complex quantum compilers.

ACKNOWLEDGMENTS

We acknowledge support from the CAM/FEDER Project No. S2018/TCS-4342 (QUITEMAD-CM), Spanish MINECO grants MINECO/FEDER Projects, PGC2018-099169-B-I00 FIS2018, MCIN with funding from European Union NextGenerationEU (PRTR-C17.I1) an Ministry of Economic Affairs Quantum ENIA project. M. A. M.-D. has been partially supported by the U.S. Army Research Office through Grant No. W911NF-14-1-0103. S.G. acknowledges support from a QUITEMAD grant. We acknowledge the precious support of R. Fazio (ICTP and Università degli studi di Napoli “Federico II”), P. Lucignano (Università degli studi di Napoli “Federico II”) and the Università degli studi di Napoli “Federico II.”

APPENDIX: POST-PROCESSING AND NORMALIZATION

As we mentioned in Sec. III A, we have to check and fix the coefficients of the quantum states after we manage to approach them with the quantum circuits generated by our algorithm. In particular, we address a post-processing procedure that allow us to tune the coefficients in the superposition, if these latter do not meet the desired ones. This procedure consists in replacing one or more Hadamard or C-Hadamard gates

of the circuit that results from the Q-learning procedure, with unitary gates, with one or more free parameters. In general, a unitary gate can be written as

$$U(\theta, \phi, \lambda) = \begin{pmatrix} \cos(\frac{\theta}{2}) & -e^{i\lambda} \sin(\frac{\theta}{2}) \\ e^{i\phi} \sin(\frac{\theta}{2}) & e^{i(\phi+\lambda)} \cos(\frac{\theta}{2}) \end{pmatrix} \quad (A1)$$

by setting the parameters θ , ϕ and λ we can build every single-qubit unitary gate. As our interest is to modify real coefficients, we need to use the U gate as a pure rotation, in order to create an *unbalanced* Hadamard gate (or C-Hadamard gate). Indeed, the Hadamard gate corresponds to $U(\frac{\pi}{2}, 0, \pi)$ and, leaving the θ parameter free, we have a pure rotation:

$$U(\theta, 0, \pi) = \begin{pmatrix} \cos(\frac{\theta}{2}) & \sin(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) & -\cos(\frac{\theta}{2}) \end{pmatrix} = U(\theta). \quad (A2)$$

Notice that in the case of the C-U gate, its matrix representation reads

$$C-U(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos(\frac{\theta}{2}) & \sin(\frac{\theta}{2}) \\ 0 & 0 & \sin(\frac{\theta}{2}) & -\cos(\frac{\theta}{2}) \end{pmatrix}. \quad (A3)$$

We assume to replace, in the output circuit of the Q-learning algorithm, m Hadamard or C-Hadamard gates, we call the i th unitary gate $U_i(\theta_i)$ or $C-U_i(\theta_i)$, where θ_i is the i th free parameter. We then apply to the initial state the circuit with the replacements, the resulting state after this quantum circuit has the same terms of the objective one in Eq. (6) but has still undetermined coefficients:

$$|\Psi\rangle_{\text{out}} = \sum_{j=1}^{16} f_j(\theta_0, \dots, \theta_m) |\psi_j\rangle. \quad (A4)$$

Here the coefficients $f_j(\theta_0, \dots, \theta_i, \dots, \theta_m)$ can assume any desired value, with the requirement of $|\Psi\rangle$ normalization. If the desired representative state of a SLOCC class reads

$$|\Psi\rangle = \sum_{j=0}^{16} \alpha_j |\psi_j\rangle, \quad (A5)$$

we can find the values of the parameters θ_i by solving the following equations system:

$$\begin{cases} f_0(\theta_0, \dots, \theta_i, \dots, \theta_m) = \alpha_0 \\ \vdots \\ f_j(\theta_0, \dots, \theta_i, \dots, \theta_m) = \alpha_j \\ \vdots \\ f_n(\theta_0, \dots, \theta_i, \dots, \theta_m) = \alpha_n \end{cases} \quad \text{where } \alpha_j \in \mathbb{R}, \forall j. \quad (A6)$$

On the left side, we have the undetermined coefficients for each term, in form of trigonometric functions of $\theta_1, \dots, \theta_m$, where m stands for the overall number of Hadamard and C-Hadamard gates replaced; on the right side we have the desired coefficients, $\alpha_0, \dots, \alpha_n$, where n is the number of terms of the goal state. Despite the existence of the solution is not theoretically guaranteed for that system, in our cases we always find suitable values for θ_i that allow us to reach the representative state $|\Psi\rangle$ of the SLOCC class in exam.

Let us take as an example the SLOCC class B1.1, belonging to the SLOCC family L_{abc_2} . With the Q-learning procedure we manage to find the circuit in Fig. 16. Taking into account the normalization coefficients of the Hadamard and C-Hadamard gates, the resulting state after this circuit reads

$$|\Psi\rangle_{\text{out}} = \frac{1}{\sqrt{2}}|0000\rangle + \frac{1}{2}|0110\rangle + \frac{1}{2}|1111\rangle \quad (A7)$$

in order to obtain a state that matches the normalized representative state of the class $|\Psi\rangle_{B1.1} = \frac{1}{\sqrt{3}}(|0001\rangle + |0110\rangle + |1011\rangle)$ we can apply our post-processing procedure and replace the first Hadamard gate $H(C)$ with a unitary gate $U(C)$. The result reads

$$|\Psi\rangle_{\text{out}} = \cos\left(\frac{\theta_0}{2}\right)|0000\rangle + \frac{\sin(\theta_0/2)}{\sqrt{2}}|0110\rangle + \frac{\sin(\theta_0/2)}{\sqrt{2}}|1111\rangle. \quad (A8)$$

It is straightforward that, in order to obtain $|\Psi\rangle_{\text{out}} = |\Psi\rangle_{B1.1}$ the system to solve is the following:

$$\begin{cases} \cos\left(\frac{\theta_0}{2}\right) = \frac{1}{\sqrt{3}} \\ \frac{\sin(\theta_0/2)}{\sqrt{2}} = \frac{1}{\sqrt{3}} \\ \frac{\sin(\theta_0/2)}{\sqrt{2}} = \frac{1}{\sqrt{3}} \end{cases} \quad (A9)$$

which solution include

$$\{\theta_0 \rightarrow 2 \arctan(\sqrt{2}) + 4\pi k \mid k \in \mathbb{Z}\}. \quad (A10)$$

The resulting state with this choice of the θ_0 parameter, is the desired one. Notice that in this case, as we replaced a single Hadamard gate, we have only one unknown. In some cases, we need to replace more than one Hadamard or C-Hadamard gate to obtain a solvable system.

[1] A. L. Samuel, Some studies in machine learning using the game of checkers, *IBM J. Res. Dev.* **3**, 210 (1959).
 [2] S. Raschka, *Python Machine Learning* (Packt, 2015).

[3] C. Szepesvári, Algorithms for Reinforcement Learning, *Synthesis Lectures On Artificial Intelligence And Machine Learning* **4**, 1 (2010).

- [4] M. Krenn, M. Erhard, and A. Zeilinger, Computer-inspired quantum experiments, *Nat. Rev. Phys.* **2**, 649 (2020).
- [5] A. A. Melnikov, H. P. Nautrup, M. Krenn, V. Dunjko, M. Tiersch, A. Zeilinger, and H. J. Briegel, Active learning machine learns to create new quantum experiments, *Proc. Nat. Acad. Sci.* **115**, 1221 (2018).
- [6] M. Krenn, M. Malik, R. Fickler, R. Lapkiewicz, and A. Zeilinger, Automated Search for New Quantum Experiments, *Phys. Rev. Lett.* **116**, 090405 (2016).
- [7] M. Krenn, A. Hochrainer, M. Lahiri, and A. Zeilinger, Entanglement by Path Identity, *Phys. Rev. Lett.* **118**, 080401 (2017).
- [8] H. J. Briegel and G. De las Cuevas, Projective Simulation for Artificial Intelligence, *Sci. Rep.* **2**, 400 (2012).
- [9] G. D. Paparo, V. Dunjko, A. Makmal, M. A. Martin-Delgado, and H. J. Briegel, Quantum Speedup for Active Learning Agents, *Phys. Rev. X* **4**, 031002 (2014).
- [10] V. Dunjko and H. J. Briegel, Machine learning & artificial intelligence in the quantum domain: A review of recent progress, *Rep. Prog. Phys.* **81**, 074001 (2018).
- [11] L. Lamata, Quantum machine learning and quantum biomimetics: A perspective, *Mach. Learn.: Sci. Technol.* **1**, 033002 (2020).
- [12] J. C. H. Watkins and P. Dayan, Q-learning, *Mach. Learn.* **8**, 279 (1992).
- [13] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, (MIT Press, 2018).
- [14] R. Bellman, On the Theory of Dynamic Programming, *Proc. Natl. Acad. Sci. USA* **38**, 716 (1952).
- [15] F. S. Melo and M. I. Ribeiro, Convergence of q-learning with linear function approximation, 2007 European Control Conference (ECC), 2671-2678 (2007).
- [16] *Dynamic Programming*, edited by R. E. Bellman, Dover Books on Computer Science Series (Dover Publications, 2003).
- [17] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, 2010).
- [18] A. Galindo and M. A. Martin-Delgado, Information and computation: Classical and quantum aspects, *Rev. Mod. Phys.* **74**, 347 (2002).
- [19] M. A. Nielsen, Conditions for a Class of Entanglement Transformations, *Phys. Rev. Lett.* **83**, 436 (1999).
- [20] F. Verstraete, J. Dehaene, and B. De Moor, Normal forms and entanglement measures for multipartite quantum states, *Phys. Rev. A* **68**, 012103 (2003).
- [21] F. Verstraete, J. Dehaene, B. De Moor, and H. Verschelde, Four qubits can be entangled in nine different ways, *Phys. Rev. A* **65**, 052112 (2002).
- [22] D. Li, X. Li, H. Huang, and X. Li, Slocc classification for nine families of four-qubits, *Quantum Inf. Comput.* **9**, 778 (2009).
- [23] L. Lamata, J. León, D. Salgado, and E. Solano, Inductive entanglement classification of four qubits under stochastic local operations and classical communication, *Phys. Rev. A* **75**, 022318 (2007).
- [24] E. Chitambar, D. Leung, L. Mancinska, M. Ozols, and A. Winter, Everything you always wanted to know about locc (but were afraid to ask), *Commun. Math. Phys.* **328**, 303 (2014).
- [25] M. Aulbach, Classification of entanglement in symmetric states, *Int. J. Quantum. Inform.* **10**, 1230004 (2012).
- [26] C. H. Bennett, H. J. Bernstein, S. Popescu, and B. Schumacher, Concentrating partial entanglement by local operations, *Phys. Rev. A* **53**, 2046 (1996).
- [27] G. Vidal, Entanglement monotones, *J. Mod. Opt.* **47**, 355 (2000).
- [28] B. Kraus, Local Unitary Equivalence of Multipartite Pure States, *Phys. Rev. Lett.* **104**, 020504 (2010).
- [29] A. Peres, *Quantum Theory: Concepts and Methods*, Fundamental Theories of Physics (Springer Netherlands, 1995).
- [30] E. Chitambar, R. Duan, and Y. Shi, Tripartite Entanglement Transformations and Tensor Rank, *Phys. Rev. Lett.* **101**, 140502 (2008).
- [31] W. Dür, G. Vidal, and J. I. Cirac, Three qubits can be entangled in two inequivalent ways, *Phys. Rev. A* **62**, 062314 (2000).
- [32] M. Swain, A. Rai, M. K. Selvan, and P. K. Panigrahi, Single photon generation and non-locality of perfect w-state, *J. Opt.* **22**, 075202 (2020).
- [33] M. O. Scully and M. S. Zubairy, *Quantum Optics* (Cambridge University Press, 1997).
- [34] V. V. Shende, S. S. Bullock, and I. L. Markov, Synthesis of Quantum Logic Circuits, *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **25**, 1000 (2006).
- [35] D. Maslov, G. W. Dueck, D. M. Miller, and C. Negrevergne, Quantum circuit simplification and level compaction, *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **27**, 436 (2008).
- [36] M. Saeedi and I. L. Markov, Synthesis and Optimization of Reversible Circuits-A Survey, *ACM Comput. Surv.* **45**, 1 (2013).
- [37] A. Bocharov, M. Roetteler, and K. M. Svore, Efficient synthesis of probabilistic quantum circuits with fallback, *Phys. Rev. A* **91**, 052317 (2015).
- [38] A. Bocharov, M. Roetteler, and K. M. Svore, Efficient Synthesis of Universal Repeat-Until-Success Quantum Circuits, *Phys. Rev. Lett.* **114**, 080502 (2015).
- [39] R. Nichols, L. Mineh, J. Rubio, J. C. F. Matthews, and P. A. Knott, Designing quantum experiments with a genetic algorithm, *Quantum Sci. Technol.* **4**, 045012 (2019).
- [40] J. M. Arrazola, T. R. Bromley, J. Izaac, C. R. Myers, K. Brádler, and N. Killoran, Machine learning method for state preparation and gate synthesis on photonic quantum computers, *Quantum Sci. Technol.* **4**, 024004 (2019).